University of Southern California

(12)

Creating and ~~Constructing~~ Coordinating Multiple Planning Methods

Soowon Lee and Paul S. Rosenbloom
USC/ Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292

June 1992
ISI/RR-93-347

INFORMATION SCIENCES INSTITUTE ISI
310/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

12

Coordinating

Creating and ~~Constructing~~ Multiple Planning Methods

Soowon Lee and Paul S. Rosenbloom
USC/ Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292

June 1992
ISI/RR-93-347

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE July 1992	3. REPORT TYPE AND DATES COVERED Research Report

4. TITLE AND SUBTITLE Creating and Coordinating Multiple Planning Methods	5. FUNDING NUMBERS N00014-89-K-0155
6. AUTHOR(S) Soowon Lee and Paul Rosenbloom	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC INFORMATION SCIENCES INSTITUTE 4676 ADMIRALTY WAY MARINA DEL REY, CA 90292-6695	8. PERFORMING ORGANIZATON REPORT NUMBER RR-347
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) ARPA 3701 Fairfax Drive Arlington, VA 22203	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Appeared in the proceedings of the 2nd Pacific Rim International Conference on AI, 1992

12A. DISTRIBUTION/AVAILABILITY STATEMENT UNCLASSIFIED/UNLIMITED	12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A planning system which has a fixed method will have trouble performing efficiently over a wide range of problems. The paper provides an alternative approach, called multi-method planning, which can potentially achieve planner completeness, planning time effciency, and plan length reduction at the same time. A way to construct multi-method planners from a set of single-method planners is introduced, and the constructed planners are compared with single-method planners. Analytical and experimental results indicate the potential of this approach.

14. SUBJECT TERMS Multi-method planning, Single-method planning. Restricted dominance graph, Monotonicity, Planning bias, Soar	15. NUMBER OF PAGES 7
	16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any avail[illegible] the public. Enter additional limita[illegible] special markings in all capitals (e.g. NOFO[illegible], ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Creating and Coordinating Multiple Planning Methods*

Soowon Lee & Paul S. Rosenbloom
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292, U.S.A.
swlee@isi.edu

(To appear in *Proceedings of the Second Pacific Rim International Conference on AI*, 1992.)

Abstract

A planning system which has a fixed method will have trouble performing efficiently over a wide range of problems. This paper provides an alternative approach, called *multi-method planning*, which can potentially achieve planner completeness, planning time efficiency, and plan length reduction at the same time. A way to construct multi-method planners from a set of single-method planners is introduced, and the constructed planners are compared with single-method planners. Analytical and experimental results indicate the potential of this approach.

Introduction

Research in domain-independent planning has been a main stream in the area of AI planning. In the design of domain-independent planning systems, it is important to consider the following criteria: (1) the ability to find a plan, or an optimal plan, for any problem in an arbitrary domain; (2) the amount of time required to find the plan; and (3) the execution cost of the plan itself. The key issue here is how to construct a single planning method, or a coordinated set of different planning methods, that has sufficient scope and efficiency.

Most planning systems encode planning behaviors within a fixed planning method such as linear planning, nonlinear planning, abstraction, and so on. Our hypothesis is that no single method will satisfy the above three criteria for all situations. For example, STRIPS-type linear planners are based on the linearity assumption to reduce the number of goal conjuncts to consider for each planning step. However, the assumption makes the planners unable to generate an optimal plan for certain problems in domains like the blocks world (Sussman, 1975) and fail to find a plan in domains with irreversible operators (Veloso, 1989). On the other hand, nonlinear planners which are free from the linearity assumption may need more effort to find a plan, because they have more choices to consider at each planning step.[1]

An alternative approach is to construct a *multi-method planner* which consists of a sequence of single-method planners, where each single-method planner has a different set of constraints. One way to determine the sequence of planners is to start with the most restricted planner and to progress on to less restricted planners, as the current one fails, until a solution is found. The idea is that if the constraints used in restricted planners can prune the search space and the eliminated space contains inefficient plans, the problems solvable by more restricted planners should be solved more quickly, generating efficient plans, while problems requiring less restricted planners should not waste too much extra time trying out the insufficient early planners. This approach is inspired by iterative deepening (Korf, 1985). In iterative deepening, a sequence of depth-first searches are performed, each to a greater depth than the previous one. If a solution is found at a shallow depth, the cost of searching to a greater depth is saved. If a solution is not found at a particular depth, a deeper search is performed. The cost of doing the shallower searches is then wasted, but since the deeper search costs at least B times the cost of the shallower search — where B is the branching factor of the search tree — this cost can be relatively quite small. Thus, if the proportion of problems solvable at shallow depths is large enough, and the ratio of costs for successive levels is large enough, there should be a net gain.

This paper describes an approach to building efficient multi-method planners from a set of single-method planners and evaluates the performance of them analytically and experimentally. We implemented both the single-method planners and the multi-method planners in the context of the Soar architecture (Laird, Newell, & Rosenbloom, 1987). Soar is a useful vehicle for this work because its impasse-driven subgoaling scheme provides the necessary context for planning and its multiple problem-space scheme provides a framework for multi-method planning. To simplify the analysis, we focus on plans represented by STRIPS-like operators.

*This work was sponsored by the Defense Advanced Research Projects Agency (DOD) and the Office of Naval Research under contract number N00014-89-K-0155.

[1]The term "nonlinear" in this context implies that operators in service of different goal conjuncts can be interleaved. It does not necessarily mean that partially ordered plans are used.

Single-method planners

A planner can be characterized by a combination of a set of *biases* over the space of plans considered and a set of control strategies that determine which plans should be considered before the others within the space. In this section, we introduce six single-method planners which are defined by different combinations of biases, as implemented in Soar, and evaluate them experimentally in terms of planner completeness, planning efficiency, and plan length. We do not focus on control strategy here.

Planning biases

With the view of planning as search over a plan space, which consists of all possible sequences of operators that lead to the goal state, bias is defined as a constraint over the space of plans considered (Rosenbloom, Lee, & Unruh, 1992). To be specific, bias is "any basis for choosing one plan over another other than the goal test". Together with the planner's input — the combination of an initial state and a goal — bias thus determines which portion of the entire plan space can or will be the output of planning. Thus, the combination of biases used in a planner characterizes the plans that can be generated.

The planning biases that we have concentrated on currently are *protection*, *linearity* — two common biases in planning — and *directness*. A protection bias eliminates all plans in which an operator undoes an initial goal conjunct that is either true a priori or established by an earlier operator in the sequence.[2] A linearity bias removes from the plan space all plans in which operators in service of different unachieved goal conjuncts occur in succession; that is, once an operator for one unachieved goal conjunct is in the plan, operators for other conjuncts can be placed only after the sequence of operators for the first goal conjunct (Fikes, Hart, & Nilsson, 1971). A directness bias eliminates all plans in which there is at least one operator that does not directly achieve a goal conjunct included in the problem definition.

These three biases have all been implemented as options within a hybrid planning system. The implemented system consists of six single-method planners defined by the cross-product of two bias dimensions — goal flexibility and goal protection. Figure 1 characterizes the 3×2 set of planning methods derived from these bias dimensions. The goal-flexibility dimension is shown along the top row of the figure. It determines the degree of flexibility the planner has in generating new subgoals and in shifting the focus in the goal hierarchy. This dimension subsumes the directness and linearity biases. The most restricted point along this dimension allows no generation of new subgoals for precondition violation. That is, if an operator has unmet preconditions when the attempt is made to incorporate it into a plan, that operator is rejected. This implements a directness bias by ensuring that each of the operators in a plan directly achieves an initial goal conjunct, rather than an unmet precondition of another operator.

[2]Other forms of protection can be found in (Sussman, 1975; Warren, 1974; Waldinger, 1975).

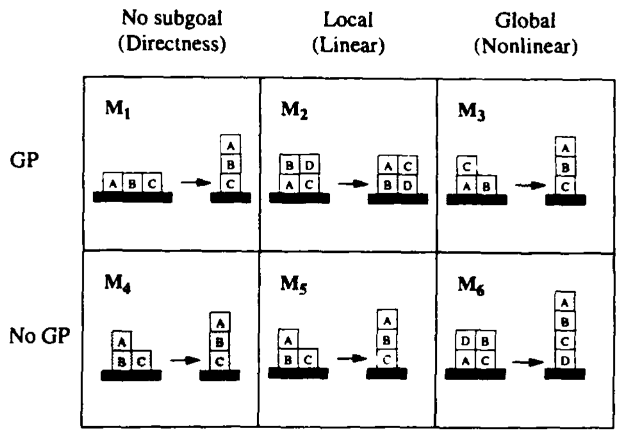

Figure 1: The planning methods generated by the bias dimensions (the bottom-left cell represents an extended blocks world problem where a block that is not clear can be moved, dropping all the blocks above it onto the table).

The second point along the flexibility dimension allows generation of new subgoals, but only a single local set of conjuncts are attended to at any point in time. Initially the local set consists of the conjuncts in the problem specification. However, whenever a selected operator has one or more unmet preconditions, the previous local set is pushed on a stack, and the operator's unmet preconditions become the new local set. When the operator's conditions are satisfied, the stack is popped to return to the previous set. Under the assumption that an operator achieves only one goal conjunct, this implements linear planning by restricting the placement of an operator within the context of the local conjuncts from which it arose, thus ensuring that operators for different goal conjuncts cannot be interleaved in the output plans.

The third point along the flexibility dimension allows the global use of subgoals; that is, new goal conjuncts are generated for unmet preconditions, and operators are simultaneously considered for all unsatisfied conjuncts. This is the least restricted version, and implements nonlinear planning by allowing operators for different goal conjuncts to be interleaved.

The goal-protection (GP) dimension is shown along the left side of Figure 1. The two points implemented along this dimension correspond to goal protection — that is, every achieved top-level goal conjunct is protected until the problem is solved — and to no goal protection. A goal protection bias shrinks the search space by cutting off sequences of operators which violate goal protection.

Each of the cells in Figure 1 shows a label representing the planner for that cell along with a problem that is just hard enough to require that planner. The most restricted planner (M_1) — a direct goal-protection planner — is in the top-left cell of the figure. While quite restrictive, it is sufficient to solve the block-stacking problem shown in that cell of the figure. The least restricted planner (M_6) — a nonlinear planner without goal protection — is in the bottom-right cell of the figure. It is the only planner in the figure capable of generating an optimal solution to the blocks-world problem shown in that cell. Between these two extremes, moving up or to the left yields more bias, while moving down or to the right yields less bias.

Implementation in Soar

The hybrid planning system containing the six single-method planners has been implemented in the context of the Soar architecture (Laird, Newell, & Rosenbloom, 1987). Problem solving in Soar is driven by applying operators to states within a problem space to achieve a goal. Knowledge is stored in a permanent recognition memory and a temporary working memory. Recognition memory consists of a set of variabilized rules, where the conditions of each rule are matched against working memory and the actions of a matched rule are instantiated to propose preferences that change the working memory. The most typical preferences are feasibility (acceptable, reject) and desirability (best, better, indifferent, worse, worst) preferences. These preferences are held in preference memory and used by a decision procedure to determine what changes are made to working memory. A subgoal is created when an impasse arises in the decision procedure. As the result of the subgoal, new preferences are generated and new rules are learned (via a chunking process) whose actions are based on the working-memory elements that are the results of the subgoal, and whose conditions are based on the working-memory elements that led to the results (Rosenbloom & Newell, 1986). In effect, chunking is much like explanation-based learning (Rosenbloom & Laird, 1986).

Figure 2 illustrates initial traces of particular versions of the single-method planners as implemented in Soar for Sussman's anomaly in the blocks world. It starts with a combination of the initial state and the entire conjunctive goal — (And (On B C) (On A B)). By means-ends analysis, it generates the set of candidate operators — (move B C) and (move A B) — that are known to potentially be able to achieve any of the goal conjuncts. A tie impasse then occurs unless there is information about how to pick among them.[3] In this tie impasse, a look-ahead search begins by selecting one of the alternatives to evaluate — here it is (move A B). Its preconditions are tested and if it is known to be applicable, it is executed. If it is not known to be applicable, what happens next depends on which biases are used in the method.

If the directness bias is used, as in Figure 2(a), the evaluation of (move A B) is terminated immediatedly, with failure as the evaluation value, and the other operator (move B C) is selected. If the directness bias is not used, a new set of goal conjuncts are generated from the operator's unmet preconditions (Figure 2(b-

[3]For simplicity of presentation, these traces only show tie impasses. Refer to (Rosenbloom, Lee, & Unruh, 1990) for other types of impasses in planning.

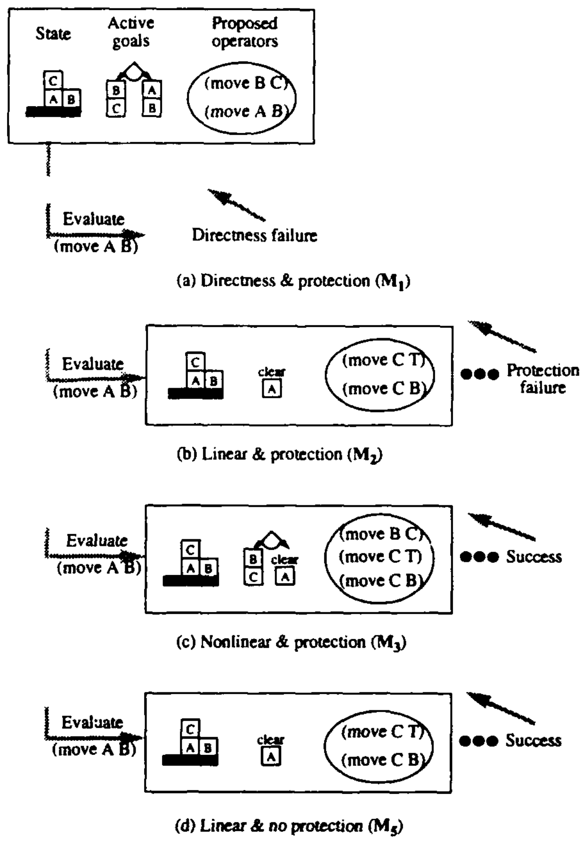

Figure 2: Planning in Soar.

d)). The difference between linear and nonlinear planning, at least for these versions, is in the focus of operator generation from the new goal hierarchy. Linear planning shifts focus completely to the new conjunct — (Clear A) as in Figure 2(b) and (d) — stays with it until it is achieved, and then pops back to the original conjunct that led to the impasse. Processing shifts to one of its siblings (if there are any) only after the original conjunct is achieved. This eventually leads to failure if a protection bias is used (Figure 2(b)), or generates a non-optimal plan if a protection bias is not used (Figure 2(d)). Nonlinear planning instead shifts to an expanded set of conjuncts that includes the new set plus the original set minus the conjunct that led to the impasse, yielding (On B C) and (Clear A) in this example (Figure 2(c)). At any point in time, an operator can be selected for any of these conjuncts, enabling operator sequences to be interleaved as necessary. For all of the above planning methods (except Figure 2(a)), once the new focus has been determined, planning continues recursively by using means-ends analysis to generate candidate operators from the new goal hierarchy.

Experimental performance of the single-method planners

Experimental results from the six planners are shown in Table 1. This data comes from running each planner on the same set of 100 problems. Because random choices are made among the goal conjuncts and among the operators proposed for evaluation, three trials are made for each problem and the results are averaged. For each problem, an initial state was randomly generated containing three or four blocks. Likewise a set of goal conjuncts was randomly generated that numbered between two and the number of blocks in the initial state. Learning was turned on for each problem, but only within-trial transfer was allowed; that is, rules learned during one problem were not used for other problems. This learning essentially enables dependency-directed backtracking and transfer across decisions for a single problem.

Table 1(a) shows the number of problems solvable in principle by each cell's planner, plus a label for the problem set that this implicitly defines. Not surprisingly, this shows a monotonic relationship between planner bias and scope, from a low of 68 problems for the most restricted planner to a high of 100 problems for the least restricted planner. Tables 1(b) and 1(c) show the average number of decisions and the average plan lengths — which should positively correlate, respectively, with planning time and execution time — for each of the four problem sets defined in Table 1(a).[4] These four problem sets are associated with the four rows within each cell. The averages in each cell only include the data from the trials that were solved within an a priori limit of 300 decisions. Since 99% of the solvable problems were actually solved within this limit, this includes nearly all of the trials.

The timing results in Table 1(b) show that planning effort is also a monotonically decreasing function of the amount of bias along these dimensions (though there is one reversal when going from M_2 to M_3 for problem set A_2). For example, for problem set A_1, effort ranged from a low of 16.3 decisions for the most biased method to a high of 39.0 decisions for the least biased method. This trade off between efficiency and completeness implies that selecting an appropriate amount of bias for a given problem is critical for finding a solution quickly.

Table 1(c) exhibits a monotonic relationship between plan length and the amount of bias used, but only for directness and protection. It shows that the linearity bias does not help here in generating shorter plans. The most likely cause of the reversal for the linearity bias is that this bias is weak enough to allow solutions to be found for all blocks world problems, but strong enough to eliminate optimal solutions for some of the problems; for example, when the shortest plan requires operators in service of different goal conjuncts to be considered before the current goal conjunct is achieved.

	No subgoal (Directness)	Local (Linear)	Global (Nonlinear)
	M_1	M_2	M_3
GP	68 (A_1)	95 (A_2)	96 (A_3)
	M_4	M_5	M_6
No GP	68 (A_4)	100 (A_5)	100 (A_6)

(a) Number of problems solvable in principle.

		No subgoal (Directness)	Local (Linear)	Global (Nonlinear)
		M_1	M_2	M_3
GP	$A_1(A_4)$	16.3	18.7	19.5
	A_2	-	28.0	27.4
	A_3	-	-	27.8
	$A_5(A_6)$	-	-	-
		M_4	M_5	M_6
No GP	$A_1(A_4)$	16.3	30.5	39.0
	A_2	-	39.1	49.9
	A_3	-	40.3	51.3
	$A_5(A_6)$	-	39.9	52.6

(b) Average number of decisions per problem solved.

		No subgoal (Directness)	Local (Linear)	Global (Nonlinear)
		M_1	M_2	M_3
GP	$A_1(A_4)$	1.82	1.84	1.89
	A_2	-	2.41	2.37
	A_3	-	-	2.39
	$A_5(A_6)$	-	-	-
		M_4	M_5	M_6
No GP	$A_1(A_4)$	1.82	3.61	3.32
	A_2	-	4.41	4.14
	A_3	-	4.57	4.17
	$A_5(A_6)$	-	4.57	4.36

(c) Average plan length per problem solved.

Table 1: Results from the six planners on three independent repetitions of a hundred randomly generated blocks world problems.

Multi-method planners

One of the main problems with the planners examined in the previous section is that each is either incomplete or performs a significant amount of excess work for some of the problems (both in planning and execution). An alternative approach is to build a ***multi-method planner*** which has a coordinated set of planning methods, where each individual planning method has a different set of constraints. A multi-method planner can be coordinated in either a sequential or a time-shared manner. A sequential multi-method planner consists of a sequence of single-method planners, while a time-shared multi-method planner consists of a set of single-method planners, where each method is active in turn for a given time slice (Barley, 1991).[5]

In this paper, sequential multi-method planning is investigated and evaluated both analytically and empirically. To simplify the analysis we focus on a special type of sequential multi-method planners called

[4]In the standard blocks world domain, M_1 is the same method as M_4. Although M_5 is a different method from M_6, and may not be able to generate an optimal plan for this domain, both M_5 and M_6 are complete planners for this blocks world domain, yielding $A_5 = A_6$.

[5]Another type of coordination for a multi-method planner would be to run the methods in parallel, as in multi-agent planning; however, the focus here is on the control of a single serial agent only.

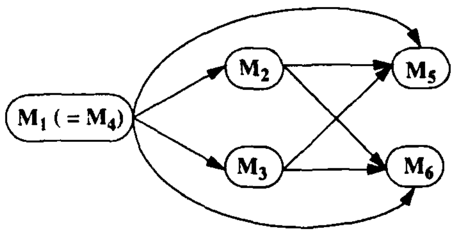

Figure 3: A restricted dominance graph for the single-method planners.

a *monotonic multi-method planner*, where the single-method planners are ordered according to increasing coverage and decreasing efficiency. That means, planning starts with the most restricted planner, and then successively relaxes the restrictions until a method is found that is sufficient for the problem. We first present a scheme to construct monotonic multi-method planners from a set of single-method planners, and then provide a formal model to compare the performance of constructed monotonic multi-method planners with single-method planners. Experimental results for them follow at the end of the section.

Constructing monotonic multi-method planners

Let M_{k_i} be a single-method planner, which can range over M_1 to M_6 defined in the previous section. A sequential multi-method planner which consists of n different single-method planners is denoted as $M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}$. Let A be a sample set of problems, and let $A_{k_i} \subseteq A$ be the subset of problems which are solvable in principle by M_{k_i}. The functions $s(M_{k_i}, A_s)$ and $l(M_{k_i}, A_s)$ represent respectively the average cost that M_{k_i} requires to succeed and the average length of plans generated by M_{k_i}, for the problems in $A_s \subseteq A_{k_i}$. Similarly, $f(M_{k_i}, A_f)$ represents the average wasted cost for M_{k_i} to fail for the problems in $A_f \subseteq A - A_{k_i}$.

A restricted dominance relation $M_x \prec M_y$ is defined between two different single-method planners, M_x and M_y, if the following conditions hold:

$$A_x \subset A_y,$$

$$s(M_x, A_{k_i}) \leq s(M_y, A_{k_i}),\ \textit{for every}\ A_{k_i} \subseteq A_x\ \textit{and}$$

$$l(M_x, A_{k_i}) \leq l(M_y, A_{k_i}),\ \textit{for every}\ A_{k_i} \subseteq A_x.$$

A sequential multi-method planner $M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}$ is called *monotonic* if $M_{k_i} \prec M_{k_{i+1}}$ holds for each $i = 1, \ldots, n-1$. Figure 3 exhibits a restricted dominance graph for the single-method planners, M_1, M_2, M_3, M_5, and M_6, which are defined in the previous section. Each node in the graph represents a single-method planner, and arc from M_x to M_y implies that $M_x \prec M_y$ holds. Thus every path in the graph corresponds to a monotonic multi-method planner. In this example, eight 2-method planners and four 3-method planners can be constructed.

Performance analysis

For each monotonic multi-method planner $M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}$, there is a corresponding single-method planner M_{k_n} which has the same coverage of solvable problems. If $M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}$ is complete, M_{k_n} is also complete. We compare a complete monotonic multi-method planner with its corresponding single-method planner in terms of planning time and plan length.

The probability that an arbitrary problem in A is solvable by M_{k_i}, which is equivalent to A_{k_i}/A, is denoted as P_{k_i}. Let M_{k_0} be a null planner which cannot solve any problem; that means $A_{k_0} = \phi$ and $P_{k_0} = 0$. Let $B_{k_i} = A_{k_i} - A_{k_{i-1}}$, for $1 \leq i \leq n$, be the set of problems which are solvable by M_{k_i} but not by $M_{k_{i-1}}$, and let $Q_{k_i} = P_{k_i} - P_{k_{i-1}}$, for $1 \leq i \leq n$.

Planning time: The expected planning time s_E of a complete monotonic multi-method planner $M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}$ can be represented as

$$s_E(M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}, A) =$$

$$\sum_{i=1}^{n}[Q_{k_i} * (s(M_{k_i}, B_{k_i}) + \sum_{j=1}^{i-1} f(M_{k_j}, B_{k_i}))], \qquad (1)$$

The performance of the corresponding single-method planner M_{k_n} is $s(M_{k_n}, A)$, which can be rewritten as the sum of the average planning time for the disjoint problem sets $A_{k_i} - A_{k_{i-1}} (1 \leq i \leq n)$:

$$\sum_{i=1}^{n}[Q_{k_i} * s(M_{k_n}, B_{k_i})]. \qquad (2)$$

To compare a monotonic multi-method planner with the corresponding single-method planner, we need to subtract (1) from (2), yielding

$$s(M_{k_n}, A) - s_E(M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}, A) =$$

$$\sum_{i=1}^{n}[Q_{k_i} * ((s(M_{k_n}, B_{k_i}) - s(M_{k_i}, B_{k_i})) - \sum_{j=1}^{i-1} f(M_{k_j}, B_{k_i}))].$$

This means that if the performance gain by using a cheaper method $(s(M_{k_n}, B_{k_i}) - s(M_{k_i}, B_{k_i}))$ is greater than the wasted time by using inappropriate methods $(\sum_{j=1}^{i-1} f(M_{k_j}, B_{k_i}))$ in a monotonic multi-method planner, then it is preferable to use that method over the single-method planner; otherwise, the single-method planner is preferred (at least where planning time is concerned).

Plan length: The estimated plan length l_E for a complete monotonic multi-method planner is:

$$l_E(M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}, A) =$$

$$\sum_{i=1}^{n}[Q_{k_i} * l(M_{k_i}, B_{k_i})],$$

while the plan length for the corresponding single-method planner M_{k_n} is

$$l(M_{k_n}, A) = \sum_{i=1}^{n}[Q_{k_i} * l(M_{k_n}, B_{k_i})].$$

If $M_{k_1} \rightarrow M_{k_2} \rightarrow \ldots \rightarrow M_{k_n}$ is monotonic, then $l(M_{k_i}, B_{k_i}) \leq l(M_{k_n}, B_{k_i})$. Therefore the lengths of plans generated from a monotonic multi-method planner are less than or equal to the length of plans generated from the corresponding single-method planner.

	Average number of decisions		Average plan length	
Planning type	expected	actual	expected	actual
$M_1 \rightarrow M_5$	39 28	35 31	3 35	3.27
$M_1 \rightarrow M_6$	46 19	42 16	3 34	3 32
$M_2 \rightarrow M_5$	35.33	35.26	2.67	2.63
$M_2 \rightarrow M_6$	37.71	37.61	2.72	2.81
$M_3 \rightarrow M_5$	33.42	33.47	2.48	2.44
$M_3 \rightarrow M_6$	35.50	34.61	2.65	2.6
$M_1 \rightarrow M_2 \rightarrow M_5$	38.52	38.00	2.66	2.76
$M_1 \rightarrow M_2 \rightarrow M_6$	40.90	38.31	2.70	2.82
$M_1 \rightarrow M_3 \rightarrow M_5$	35.99	34.09	2.43	2.42
$M_1 \rightarrow M_3 \rightarrow M_6$	38.07	35.95	2.60	2.57
M_5	-	39.94	-	4.57
M_6	-	52.61	-	4.36

Table 2· Expected and actual (experimental) performance for the monotonic multi-method planners

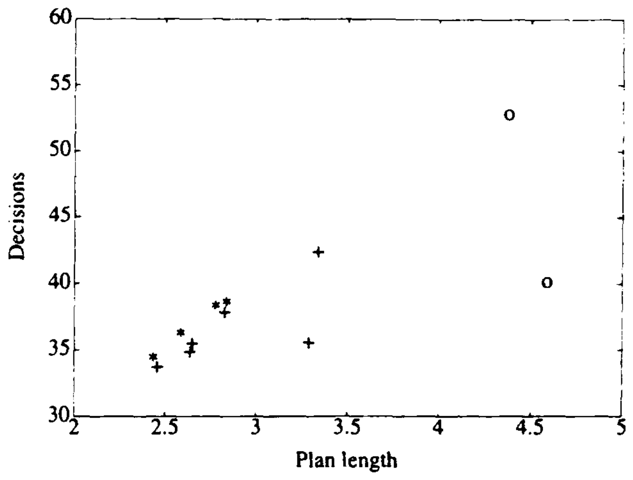

Figure 4: Monotonic multi-method vs. single-method planners ("o" represents single-method planners, "+" represents 2-method planners, and "*" represents 3-method planners.)

Experimental Results

Equation (1) makes it possible to predict the performance of monotonic multi-method planners on some problem sets. Since M_5 and M_6 are the two complete single-method planners in this domain, ten complete monotonic multi-method planners can be created from the restricted dominance graph in Figure 3: six 2-method planners and four 3-method planners. Table 2 shows the average expected number of decisions and the average expected plan length for these ten planners.

In order to validate the predictions of the model, we have implemented these ten planners in Soar. Each single-method planner in a monotonic multi-method planner was implemented as a specialization of a general problem-space. Based on the sequence of single-method planners, a set of meta-level control rules was provided to coordinate which problem-space is tried next if the current problem-space does not generate a plan for the given problem. Three repetitions were made for each problem in the same 100 problem set used in Figure 1. Within-trial learning was turned on for each problem as in the experiments with the single-method planners, but learned rules were also allowed to transfer from an earlier method to a later method (for the same problem). This is equivalent to the type of transfer allowed in the single-method planners.

The experimental results for decision cycles and plan length are presented in Table 2. It shows that the experimental decisions are slightly less than the corresponding expected decisions, while the experimental plan lengths are quite comparable to the expected plan lengths. The difference between expected decisions and experimental decisions is probably due to the across-method transfer of learned rules; that is, if two methods within a monotonic multi-method planner have the same bias and the control rules learned from the earlier method depend on only that bias, then those rules can transfer to the later method. If this is the case, then across method transfer is saving about 10%.

Figure 4 compares the ten monotonic multi-method planners and the two single-method planners based on the data in Table 2. It shows that nine out of ten multi-method planners ($M_1 \rightarrow M_6$ is the exception) outperform the two single-method planners in terms of both number of decisions and plan length for this set of biases and problems. Among the multi-method planners, 3-method planners tend to generate marginally shorter plans than 2-method planners, with a slightly increased number of decisions (except the case of $M_1 \rightarrow M_6$).

Related Work

The basic approach of bias relaxation in multi-method planning is similar to the shift of bias for inductive concept learning (Russell & Grosof, 1987; Utgoff, 1986). In the planning literature, this approach is closely related to an *ordering modification* which is a control strategy to prefer exploring some plans before others (Gratch & DeJong, 1990). Bhatnagar & Mostow (1990) described a relaxation mechanism for overgeneral censors in FAILSAFE-2. However, there are a number of differences, such as the type of constraints used, the granularity at which censors are relaxed, and the way censors are relaxed. SteppingStone (Ruby & Kibler, 1991) tries constrained search first, and moves on to unconstrained search, if the constrained search reaches an impasse (within the boundary of ordered subgoals) and the knowledge stored in memory cannot resolve the impasse.

Conclusions

In this paper, we investigated a sequential multi-method planning approach that can improve the three planning criteria: planner completeness, planning efficiency, and plan length. A formal analysis shows that (1) a monotonic multi-method planner takes less planning time then the corresponding single-method planner, if the performance gain by using a cheaper method is greater than the wasted time by using inappropriate methods in the monotonic multi-method planner;

and (2) the lengths of plans generated from a monotonic multi-method planner are less than or equal to the length of plans generated from the corresponding single-method planner. The experimental results obtained so far in the blocks world are consonant with this model (though there is a small confound due to learning).

The findings in this paper do not necessarily mean that, for all situations, there exists a monotonic multi-method planner which outperforms the most efficient single-method planner. In fact, the performance of these planners depends on the biases used in the multi-method planners and the problem set used in the experiments. For example, if the problems are so complex that most of the problems are solvable only by the least restricted method, the performance loss by trying inappropriate earlier methods in sequential multi-method planners would be critical. On the other hand, if the problems are so trivial that it takes only a few decisions for the least restricted method to solve the problems, the slight performance gain by using more restricted methods in sequential multi-method planners might be overridden by the complexity of the meta-level processing required to coordinate the sequence of primitive planners.

It thus remains an open question as to the range of situations in which multi-method planners will actually perform better. However, one way to increase the chances of multi-method planner's performing better is to take advantage of the novel optimization opportunities that they provide. One possibility is to extend the scope of the biases to ones that limit the size of the goal hierarchy to reduce the search space, limit the length of plans generated to shorten execution time, and result in learning more effective rules to increase transfer (Etzioni, 1990). Another possibility is to learn which planning method to use for which class of problems. This can reduce the time wasted by multi-method planners. Although our current implementation has the capability to learn rules that select appropriate methods, the effect of method selection on the overall system performance has not yet been fully investigated. The third possibility is to reduce the granularity at which the individual planning methods are selected and used. This means that a planning method can be switched at a subgoal selection point or an operator selection point, if there is no path from that point to reach the goal state. This approach can potentially improve the performance of multi-method planning if, for example, there are a significant number of problems where most of the subgoals are solvable by a very cheap method while the remainder of the problem requires a more complex method.

References

Barley, M. (1991). Personal Communication.

Bhatnagar, N., & Mostow, J. (1990). Adaptive search by explanation-based learning of heuristic censors. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 895-901). Boston, MA: AAAI Press.

Etzioni, O. (1990). Why Prodigy/EBL works. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 916-922). Boston, MA: AAAI Press.

Fikes, R. E., & Nilsson N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence, 2*, 189-208.

Gratch, J. M., & DeJong, G. F. (1990). A framework for evaluating search control strategies. *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 337-347). San Diego, CA: Morgan Kaufmann.

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence, 27*, 97-109.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gill, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence, 40*, 63-118.

Rosenbloom, P. S., & Laird, J. E. (1986). Mapping explanation-based generalization onto Soar. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 561-567). Philadelphia, PA: Morgan Kaufmann.

Rosenbloom, P. S., Lee, S., & Unruh, A. (1990). Responding to impasses in memory-driven behavior: A framework for planning. *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling, and Control* (pp. 181-191). San Diego, CA: Morgan Kaufmann.

Rosenbloom, P. S., Lee, S., & Unruh, A. (1992). Bias in planning and explanation-based learning. S. Chipman & A. Meyrowitz (Eds.) *Machine Learning: Induction, Analogy and Discovery*. Hingham, MA: Kluwer Academic Publishers. In Press. (Also available in S. Minton (Ed.) *Machine Learning Methods for Planning and Scheduling*. San Mateo, CA: Morgan Kaufmann. In Press.)

Russell, S. J., & Grosof, B. N. (1987). A declarative approach to bias in concept learning. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 505-510). Seattle, WA: Morgan Kaufmann.

Sussman, G. J. (1975). *A Computer Model of Skill Acquisition*. Cambridge, MA: MIT Press.

Utgoff, P. E. (1986). Shift of bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.) *Machine Learning: An Artificial Intelligence Approach, Vol. II*. Los Altos, CA: Morgan Kaufmann.

Veloso, M. (1989). *Nonlinear problem solving using intelligent casual-commitment* (Technical Report CMU-CS-89-210). School of Computer Science, Carnegie Mellon University.

Waldinger, R. (1975). *Achieving several goals simultaneously* (SRI AI Center Technical Note 107). SRI, Menlo Park, CA.

Warren, D.H.D. (1974). *WARPLAN: a system for generating plans* (Dept. of Computational Logic Memo 76) Artificial Intelligence, Edinburgh University.